31-03-2015

# Open Call Deliverable OCO-DS1.1 Network Coding in Transport Networks (MINERVA)

**Open Call Deliverable OCO-DS1.1**

| | |
|---|---|
| Grant Agreement No.: | 605243 |
| Activity: | NA1 |
| Task Item: | 10 |
| Nature of Deliverable: | R (Report) |
| Dissemination Level: | PU (Public) |
| Lead Partner: | BME |
| Document Code: | GN3PLUS14-1296-40 |
| **Authors:** | from BME: Dr. Péter Babarczi, Dr. János Tapolcai, Bence Ladóczki, Alija Pašić; from i2CAT: Carolina Fernandez, Oscar Moya Gomez, Dani Guija |


The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 605243 (GN3plus).

**Abstract**

The purpose of this document is to summarize the theoretical results, implementation considerations, deployment details, and measurement results reached in the MINERVA project. During our theoretical work we identified a special case of network coding which is able to maintain scalability and simplicity, and through our implementation we showed that it is a viable approach in transport networks.

# Table of Contents

# Table of Figures

# Table of Tables

# Executive Summary

MINERVA is the beneficiary of the “High-Availability Networking” Open Call. The objectives of the Open Call were:

- To explore and research high-availability methods/technologies.
- To port one or more applications to multi-instance versions for high-availability demonstrators.

As part of the MINERVA project, we proposed a single-link failure-resilient routing architecture (***Resilient Flow Decomposition, RFD***) for unicast connections in transport networks based on network coding. Our main goal was to maintain simplicity and scalability in the design of our resiliency method. As part of our theoretical work in Task 1.1, we proved that simple XOR coding at the source and destination nodes are sufficient to reach all benefits that intra-session network coding can offer for transport networks. We proposed a polynomial time algorithm for optimal coding graph selection, as well as simple linear time network code construction approaches, which replaced the previously involved coding schemes that are known from the literature. With these results, we satisfied the goals of the open call, i.e., ease of deployment, scalability and maintainability.

Our proposed research work covered two other aspects of high-availability networking. First, in our monograph and research papers we have thoroughly investigated all-optical failure localization; which showed that fast failure localization could be a counterpart of RFD in application scenarios where, in the recovery time – resource efficiency trade-off, the resource efficiency (i.e. usage of the channel or link) is crucial. Although signalling can be completely eliminated from the recovery process with our proposed approaches, the time of a single switching matrix configuration is still required; clearly not satisfying the instantaneous recovery offered by RFD. Second, as RFD breaks with the traditional single-path routing concepts, we have investigated future routing and forwarding architectures to support RFD-based forwarding in future networks. With all this research work, we have addressed most of the topics of interest of the open call, e.g., using routing protocols to signal instance availabilities, support for continuous replication of data, support for seamless and rapid handover and recovery, support for forgiving and self-repairing synchronisation mechanisms, etc.

In order to satisfy the second objective of the open call (i.e., applications for high-availability demonstrators), and based on the knowledge of RFD, we have developed and implemented use cases for video streaming and distributed storage. Both use cases are built on the resilient routing architecture offered by RFD. That is, video streams can be recovered from single-link failures of the transport network without any interruption, and the redundancy offered by simple network coding is enough to recover files in a distributed storage even in the presence of single link or storage node failure. Through its deployment in the GÉANT OpenFlow Facility (GOFF), we have shown that the RFD method can ***instantaneously recover*** video streams sent through an insecure protocol (e.g., UDP) from single-link failures without packet retransmission or flow rerouting. This means that *RFD+UDP can be a viable approach to offer reliable services like TCP in future transport networks*.

Among the expected outputs of the Open Call, publication of the project's results in peer-reviewed journals was an important requirement. Besides 1 book (monograph), 4 conference publications to top venues, and 1 demo paper, we have summarized our research results in 3 peer-reviewed journal papers in prestigious journals: two in ***IEEE/ACM Transactions on Networking*** and one in ***Elsevier Computer Networks***. We have also submitted 1 journal paper about optimal network coding schemes for RFD in ***IEEE Transactions on Information Theory***, which is still under review at the end of the project. The summary of the proposed and reached publications can be found in Table 1.

| | Proposed | Accepted | Submitted |
|---|---|---|---|
| Publications of T 1.1.<br>(Nov 2013 – Apr 2014) | 2 submitted conference papers | 2 journals (IEEE/ACM Transactions on Networking)<br>1 book (Springer)<br>3 conference papers (IEEE ISIT, IEEE DRCN, IEEE Infocom) | 1 journal paper (IEEE Transactions on Information Theory) |
| Publications of T 1.4.<br>(Dec 2014 – March 2015) | 1 accepted conference paper<br>1 accepted journal paper | 1 journal paper (Elsevier Computer Networks)<br>1 conference publication (IFIP Networking)<br>1 demo paper (IEEE Infocom) | - |
| Dissemination of T 2.2. | - | CONNECT Magazine | - |

Table 1: Summary of the publication activity of the MINERVA project

# 1 Introduction

A survivable routing scheme in transport networks has three utmost important features: ***recovery time, simplicity*** (i.e., low computational and operational complexity) and ***efficient capacity allocation***. But which one is the most important for service providers, and what are they willing to sacrifice in order to reach it? To answer this, we have to look to what is used in practice. In networks, the most commonly used survivable routing scheme is the so-called *dedicated 1+1 path protection*, which sends the user data along two disjoint paths (primary and backup). Although it consumes twice as much capacity as the primary capacity, there are efficient algorithms to calculate 1+1 routing (i.e., disjoint path-pair), and it provides instantaneous recovery from any single edge failure.

Although 1+1 is still the most commonly used protection scheme, there is still room for improvement in bandwidth utilization. On the other hand, and based on the previous observation, ***simplicity and ultra-fast recovery time seem to be the most important features***; and efficient capacity allocation comes only after them. Several survivable routing schemes were introduced in the past decades. Such schemes could significantly reduce the bandwidth utilization, but they sacrifice the *ultra-fast recovery time, the low computational complexity, or - most importantly - the simple operation*. Following this argument we show in MINERVA that, through a careful design, we can keep these merits and achieve near-optimal capacity allocation at the same time.

In transport networks, optimal capacity efficiency can be achieved with in-network modification of data (called ***network coding***) while *fast recovery time is maintained*. However, the practical implementation of complex network coding operations is still an open question, which makes these methods inapplicable. On the other hand, there are some special cases of coding that could be feasible in transport networks. For instance, *diversity coding* (DC); which splits the data at the source node into two parts *A* and *B*, and creates redundancy data *A XOR B*, too (XOR denotes the exclusive OR operation on the data), then sends these on three edge-disjoint paths. Diversity coding can reduce the capacity consumption of 1+1 (from 2 to 1.5 units), while its complexity and recovery time are the same. On the other hand, this method is applicable only in 3 edge-connected networks, which is a crucial drawback.

In the MINERVA Open Call project from November 2013 to March 2015 our goal was to develop a simple, scalable and easily maintainable survivable routing scheme using network coding that maintains the merits of DC and also improves it in several aspects. These results, among them the design of a reliable underlying networking infrastructure, were published in the publications listed below. We also provide a brief introduction of the contents of each of them.

The two utmost important theoretical features required for a robust network coding scheme to be considered a viable approach in practice were investigated in the first two papers. The first feature to address is *finding an optimal coding subgraph* for the connection, which ensures that even if a failure occurs, the reserved resources are sufficient to ensure the connectivity and minimal data rate between the source and destination nodes. We answered this question in the following paper (detailed in Section 2.1):

- Alija Pasic, János Tapolcai, Péter Babarczi, Erika R. Bérczi-Kovács, Zoltán Király, Lajos Rónyai, *Survivable Routing Meets Diversity Coding*, submitted to **IFIP Networking**, pp. 1-9, Toulouse, France, 2015.

The other important feature of a network coding scheme is to *find an optimal network code* in an (optimal) coding subgraph. We proposed a simple and elegant proof for this issue in the following paper:

- Péter Babarczi, János Tapolcai, Lajos Rónyai, and Muriel Médard, *Resilient Flow Decomposition of Unicast Connections with Network Coding*, in Proceedings of the **IEEE International Symposium on Information Theory (ISIT)**, pp. 116-120, Honolulu, HI, USA, 2014.

Since then, we have further investigated its scope and proposed further coding schemes (summarized in Section 2.2):

- Péter Babarczi, János Tapolcai, Alija Pasic, Lajos Rónyai, Erika R. Bérczi-Kovács, and Muriel Médard, *Linear Time Coding Algorithms for Resilient Flow Decomposition in Transport Networks*, submitted to **IEEE Transactions on Information Theory**, 2015.

Owing to its graph theoretical properties, we named the above framework (which finds an optimal coding subgraph and an appropriate network code) as ***Resilient Flow Decomposition (RFD)***. After we identified the necessary functions to make the RFD scheme work in practice, we published the architecture and the initial measurement results measured on our proof-of-concept implementation in the GÉANT OpenFlow Facility in the following paper (refer to Section 3 for further details):

- Péter Babarczi, Alija Pasic, János Tapolcai, Felicián Németh, and Bence Ladóczki, *Instantaneous recovery of unicast connections in transport networks: Routing versus*

*coding,* accepted to **Elsevier Computer Networks (COMNET)**, impact factor 1.282 (in 2013), 2015.

Finally, we have implemented two use cases on the basis of RFD, namely video streaming and distributed storage. Their demonstration is to be done between April 27th and 30th, after the project ends in March; and will take place at the demo session of IEEE Infocom (described in Section 4):

- Bence Ladóczki, Carolina Fernandez, Oscar Moya, Péter Babarczi, János Tapolcai, Daniel Guija, *Robust Network Coding in Transport Networks*, accepted to The 34rd Annual **IEEE International Conference on Computer Communications (INFOCOM)**, pp. 1-2, Hong Kong

Note that, the goal of RFD was to keep the instantaneous recovery property of 1+1 protection while offering improved resource efficiency. Another viable resiliency approach is to keep the resource efficiency of restoration methods and improve their recovery time; that is, approaching the resource efficiency – recovery time trade-off from the other extreme. This approach is applicable in environments where the resource efficiency is more important than instantaneous recovery. We have investigated some related topics in the field of high-availability networking such as fast all-optical failure localization methods for rapid recovery of the disrupted connection:

- János Tapolcai, Pin-Han Ho, Péter Babarczi, and Lajos Rónyai, *Internet Optical Infrastructure - Issues on Monitoring and Failure Restoration*, pp. 1-212, Publisher: **Springer**, ISBN 978-1-4614-7737-2, 2014.

- János Tapolcai, Pin-Han Ho, Péter Babarczi, and Lajos Rónyai, *Neighborhood Failure Localization in All-Optical Networks via Monitoring Trails*, accepted to **IEEE/ACM Transactions on Networking (ToN)**, impact factor 1.986 (in 2013), 2014.

- János Tapolcai, Lajos Rónyai, Éva Hosszu, Pin-Han Ho, Suresh Subramaniam, *Signaling Free Localization of Node Failures in All-Optical Networks*, **The 33rd Annual IEEE International Conference on Computer Communications (INFOCOM)**, pp. 1860-1868, Toronto, ON, Canada, 2014.

These approaches can be incorporated with pre-planned restoration methods (often termed as shared protection in the literature), and could provide fast recovery by eliminating both failure localization time and failure notification time form the standard GMPLS recovery process. We have shown that the additional capacity reserved by the supervisory lightpaths (purely used for failure localization purposes) can be completely hidden by the protection capacity of the restoration framework, even in lightly loaded networks. Although resource efficient, the time of a single switching matrix configuration is still required, thus, they could not provide instantaneous recovery as effectively as RFD does.

Furthermore, as RFD breaks with the traditional single-path forwarding paradigm, new forwarding and routing architectures are required which are able to support multipath routing, and enable novel forwarding and routing schemes for multicast routing and routing on directed acyclic graphs (DAGs) in RFD:

- János Tapolcai, József Bíró, Péter Babarczi, András Gulyás, Zalán Heszberger, and Dirk Trossen, *Optimal False-Positive-Free Bloom Filter Design for Scalable Multicast Forwarding*, accepted to **IEEE/ACM Transactions on Networking (ToN)**, impact factor 1.986 (in 2013), 2014.

- Máté Nagy, János Tapolcai and Gábor Rétvári, *On the Design of Resilient IP Overlays*, In Proc. **10th International Conference on Design of Reliable Communication Networks (DRCN)**, pp. 1-8, Ghent, Belgium, 2014.

The detailed description of these results on all-optical failure localization and Bloom filter based forwarding can be found in Milestone 1.1 and in the related publications. In the rest of this document, we shortly describe the theoretical basis of RFD (in Section 2) as our novel robust network coding based solution for transport networks (results of Task 1.1). After that, we describe the necessary network functions that are required to deploy an arbitrary coding subgraph in a software-defined network in Section 3 (results of Task 1.2). Section 4 introduces the two application scenarios built on the survivable routing architecture of RFD (approaches in Task 1.3) and contains some measurement results and the overall lessons learned in MINERVA (conducted during Task 1.4). Finally, Section 5 concludes the deliverable.

# 2 Theoretical results in MINERVA

In this section, we shortly summarize the main findings of the theoretical results of MINERVA, i.e., the idea behind the robust network coding architecture of the Resilient Flow Decomposition (RFD) algorithm and the main theorems and definitions. Based on these results, in this section we identify the main characteristics and node capabilities of an arbitrary RFD solution; which will be the basis of the Network Function (NF) design in Section 3. A more detailed description on the mathematical background of RFD can be found in Milestone 1.1 and in [Pasic15], [Babarczi14], [Babarczi15-TIT].

## 2.1 The MINERVA survivable routing framework

A transport network is a collection of routers, switches (referred to as nodes) and high bandwidth communication links (referred to as edges) between them. It may be represented by a directed graph *G = (V,E,k,c)* with node set *V* and edge set *E*. Each $e \in E$ edge has two attributes, namely its capacity $k(e) \in N$, i.e., number of bandwidth units available for data transmission; and its cost $c(e) \in R^+$, which is defined as the cost of using one unit of bandwidth along edge *e*. Given a connection *D=(s,t,d)*, with information source $s \in V$, information sink $t \in V$ and the number of bandwidth units *d* requested for data transmission.

> **Definition 1.** We say that $R=(V^R, E^R, f)$ is a *survivable routing* of a connection *D=(s,t,d)* in *G* (where $V^R \subseteq V$, $E^R \subseteq E$, and $\forall e \in E^R : f(e) \leq k(e)$, if there is an *s-t* flow of value $F \geq d$ in *R*, even if we delete any single edge of *R*. On the other hand, a routing is *vulnerable* if it is not survivable.

Note that, after the edge failure is identified, any routing method could be adopted to resend the flows on the intact edges of a survivable routing *R*, clearly resulting in *slow recovery*. However, several (robust) network coding theorems ensure that a sufficient amount of information reaches the destination after a failure occurred when there is no failure identification, but for the price of complex in-network operations.

***In MINERVA we have demonstrated that these merits –fast recovery and simplicity– can be brought together in survivable routing approaches with the help of our network coding approach, called Resilient Flow Decomposition (RFD).***

We say that a ***survivable routing R is critical***, if we cannot further decrease the flow value *f(e)* along any edge in $\forall e \in E^R$ without making routing *R* vulnerable. A (robust) network coding based routing scheme (like RFD) consist of two steps:

(i) *optimal coding subgraph* selection,

(ii) optimal (robust) network code construction in the coding subgraph.

Although the first step is the selection of a logically optimal coding graph, we start with the introduction of the main code construction theorem of the MINERVA framework; as this will give us a really efficient tool for designing optimal coding subgraphs as well. This result greatly simplifies the previously known code construction methods in the literature, and ***opens the way for MINERVA to be the first efficient implementation of a robust network coding based scheme in transport networks***.

## 2.2 Finding an optimal network code in an optimal coding subgraph (Resilient Flow Decomposition)

In survivable routing, besides theoretically good properties such as low bandwidth utilization and fast recovery, both simplicity and easy deployment are essential from a practical point of view. Thus, complex data processing at core nodes (i.e., other than *s* and *t*) like network coding is not desired. All complex operations shall be moved to the edge of the network (i.e., nodes *s* and *t*). Luckily, for single edge failures and two data parts, the number of survivable routings providing this simplicity is quite wide [Babarczi15-TIT], [Babarczi14]:

> **Theorem 1 (Resilient Flow Decomposition).** Suppose that survivable routing *R* is critical. Then there are disjoint edge sets $E^A$, $E^B$, $E^{A \text{ XOR } B}$ of *R*, called ***routing DAGs***, such that for an arbitrary edge $e \in E^R$ after removing the corresponding edge(s) from $E^R$ at least two of the routing DAGs connect *s* to *t*.

Figure 1 depicts an example of how a critical survivable routing can be decomposed into the three routing DAGs.

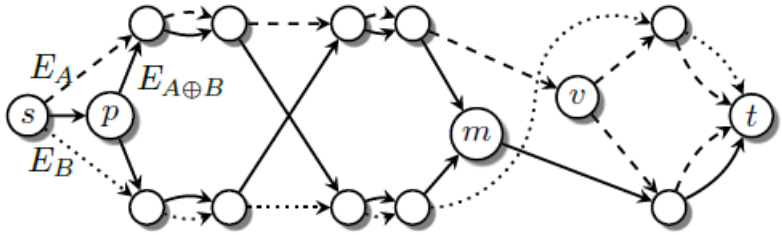


Figure 1: Example of routing DAG decomposition of a critical survivable routing

Note that, in *diversity coding*, for a connection *D=(s,t,2)* the redundant data *A XOR B* is calculated at the source *s*. Then, *A*, *B* and *A XOR B* are sent along three disjoint end-to-end paths between *s* and *t*. The edge sets used by the disjoint paths are denoted as $E^A$, $E^B$, $E^{A\ XOR\ B}$, respectively. Now, *1+1* protection scheme could be treated as an implementation of RFD: *A* and *B* are sent along two disjoint paths $E^A$ and $E^B$, while the redundant data *A XOR B* is sent along both paths, i.e., $E^{A\ XOR\ B} = E^A\ U\ E^B$. Both routings are survivable.

It is also worth noting that, with the help of RFD, the three routing DAGs can be operated independently from each other, i.e., ***each DAG carries a specific data part regardless of a failure condition; while involved operations are performed only at the end nodes of the connection***. However, the general implementation of RFD (other than diversity coding and 1+1) requires splitting and merging of the paths at the core nodes (e.g., nodes *p* and *m*, respectively in Figure 1). Let $\delta^-(v)$ and $\delta^+(v)$ denote the in-degree and the out-degree of a node, respectively.

> **Definition 2.** Node $p \in V^R$ is called ***splitter***, if $\delta^-(p) = 1$ and $\delta^+(p) = 2$, i.e., it receives data on a single edge, while forwarding the same copy on two outgoing edges. Similarly, node $m \in V^R$ is called ***merger***, if $\delta^-(m) = 2$ and $\delta^+(m) = 1$, i.e., it receives *the same data* on two incoming edges, while forwarding one of them (or upon failure the intact one) on its single outgoing edge.

We believe that splitting and merging operations are simple enough, in the sense that every node in the network can perform them without any complicated software update. Furthermore, in current networking paradigms, such as Software-Defined Networking (SDN), a splitter can be easily deployed by applying simple flow rules; while merger functionality can be implemented as a simple network function. This is further discussed in Section 3.

The corollary of Theorem 1 is that a ***robust network code in our single link failure resilient setting –when user data is split into two parts– is equivalent to find three routing DAGs between s and t*** in the coding subgraph. Thus, we have converted a complex algebraic problem to a graph theoretical (more precisely flow) problem. In Milestone 1.1 we have introduced a linear time code construction algorithm for a survivable network [Babarczi14], and we have shown that if we are considering critical survivable routing $R^D$,

than a more efficient coding exists [Babarczi15-TIT]. However, note that if we are considering optimal coding graph selection and network code construction as a joint optimization problem (i.e., find minimal cost coding subgraphs in the form of three routing DAGs); then the RFD theorem gives us the opportunity to find the optimal coding subgraph together with the corresponding network code without any subsequent network code construction step in some special cases. Such subgraph selection algorithms and the complexity of the other optimization problems will be discussed in Section 2.3.

## 2.3 Finding an optimal coding subgraph for RFD

In the optimal coding subgraph selection problem, our goal is to find a survivable routing *R* for connection *D* with minimum *bandwidth cost* from the possible set of survivable routings $R^D$. In Milestone 1.1 optimal capacity allocation for RFD was still stated as an open question. In our subsequent research, we have proved the complexity of two practically relevant scenarios. First, we have shown that the this problem is solvable in polynomial time, if the edge capacities *k(e)* are *infinite*, i.e., there are no bottleneck links in the network and all nodes can split or merge the flow. Note that for the demand *D=(s,t,2)* in RFD we are searching three routing DAGs, each forwarding one unit of capacity (either *A*, *B* or *A XOR B*. Thus, in a critical survivable routing one would think that infinite edge capacity means $\forall e \in E : k(e) = 3$, i.e., the case when all DAGs may use the same edge. However, and as a consequence of the RFD theorem, it was shown in [Babarczi14] that the flow values are $\forall e \in E : k(e) \leq 2$ for a connection with *d=2 and* using resilient flow decomposition in a *critical survivable routing*. Thus, without loss of generality, we can restrict the available capacities to $k(e) = 2$ for every edge *e*, without ruling out the minimum cost survivable routing (which is critical, obviously). Hence, infinite edge capacities in RFD mean $\forall e \in E : k(e) = 2$ (instead of 3). Furthermore, we assume first that each node in the topology can act as a splitter or merger. Based on these assumptions, in [Pasic15] we proved the following theorem:

> **Theorem 2.** If the network has infinite edge capacities on all edges (and each node can act as a splitter and/or merger), the minimum cost survivable routing *R* can be computed in $O(|V||E| log_{1+|E|/|V|} |V|)$ steps.

The proof is constructive, and gives the ***polynomial-time algorithm to find an optimal survivable routing***, detailed in [Pasic15]. A natural question is why the algorithm cannot cope with networks with some edge capacities $k(e) = 1$, or when some nodes are not able to perform splitting or merging. The problem is that in the capacity-constrained case, the route of the third routing DAG depends on the first two routing DAGs; as they can reserve some critical resource (i.e., we cannot calculate the DAGs on the same graph as the available link set changes upon resource availability). The problem of the scenario with restricted node capabilities is that splitting and duplicating the same data flow on a single link is beneficial in some scenarios and obviously could not be an issue in Theorem 2, where such a solution would lead to sub-optimality. Thus, the computational complexity of these cases is an open

question. However, the other extreme of the polynomial time special case, i.e., when ***both capacity and node capabilities are restricted, was proved to be NP-complete*** in [Pasic15].

Table 2 summarizes the complexity results of the proposed optimal algorithms with different available node sets $P$ and $M$ for splitting and merging, respectively (infinite edge capacities mean $\forall e \in E : k(e) = 2$, while capacity constrained refers to the case when there are some links with capacity $k(e) = 1$).

| | Infinite capacities | Capacity constrained |
|---|---|---|
| $P = \{s\}, M = \{t\}$ | Suurballe<br>$O(\lvert V \rvert \log_2 \lvert V \rvert + \lvert E \rvert)$ | Suurballe<br>$O(\lvert V \rvert \log_2 \lvert V \rvert + \lvert E \rvert)$ |
| $P \subseteq V, M \subseteq V$ | - | -<br>NP-complete |
| $P = M = V$ | SRDC algorithm (Survivable routing with diversity coding)<br>$O(\lvert V \rvert \lvert E \rvert \log_{1+\lvert E \rvert / \lvert V \rvert} \lvert V \rvert)$ | Integer Linear Program<br>- |

Table 2: Complexity results of other optimal survivable routing finding algorithms

## 2.4 Scope of Resilient Flow Decomposition

The linear time coding algorithms and polynomial time subgraph selection algorithms presented in this section proved that the RFD approach works effectively for the case of ***two data flows and a single link failure***. Two straightforward generalizations may increase the number of data flows or the number of possible link failures. We presented examples to show that none of these generalizations are possible, i.e., such flow decomposition algorithm may not exit for these scenarios. To be more specific, in Milestone 1.1 and in [Babarczi15-TIT] we have shown that such simple flow decomposition is not possible neither for dual link failure resilience with two data parts, nor for single link failure with three or more data parts. Therefore, we can claim that the simplicity of RFD is only proved for the investigated special case. Luckily, this is more than enough, as it ***opens the way for an efficient implementation of robust network coding in transport networks***.

# 3 The MINERVA architecture for RFD

In order to investigate the possibility of the practical deployment of Resilient Flow Decomposition (RFD), it was necessary to implement the modules that allow forming an arbitrary protection structure for the connection requests. In this section we shortly describe our proof-of-concept RFD implementation in a real-world transport network infrastructure, namely the OpenFlow Facility of the GÉANT European backbone network (GOFF). For a detailed description, please, refer to Milestone 1.2 and [Babarczi15-COMNET].

The existing implementations of network coding fall into two categories: they either put the coding/decoding functionality into the application layer of the end hosts, or they needed to substantially modify the software running in routers. Relying on a modified end host requires end user involvement, which limits the possible deployment scenarios. Additionally, relying solely on end host also reduces the number of coding possibilities. On the other hand, persuading equipment vendors to implement an experimental network coding function in their high-speed router is slow process that most probably will not succeed. Luckily, the *Software-Defined Networking (SDN)* paradigm has been gaining acceptance in recent years. SDN makes fast innovation possible by separating the control plane from the data plane. A control application (controller) can remotely “program” the packet forwarding behaviour of an SDN capable switch. Unfortunately, though, the currently standardized protocol for controller-switch communication named OpenFlow is not capable of instructing switches to modify packet payloads. However, the OpenFlow protocol conveniently allows steering traffic to middleboxes where complex packet manipulations can take place. Moreover, the advance in hardware technology now allows porting middlebox logic implemented on specialized hardware elements to normal software running on off-the-shelf hardware. In case of Network Function Virtualization (NFV), the network operator runs this packet processing software in a virtualized environment, for example, in a virtual machine. NFV allows dynamically placing so-called network functions next to different switches based on actual demands. Thus, ***we decided to follow the NFV approach in our network coding implementation***.

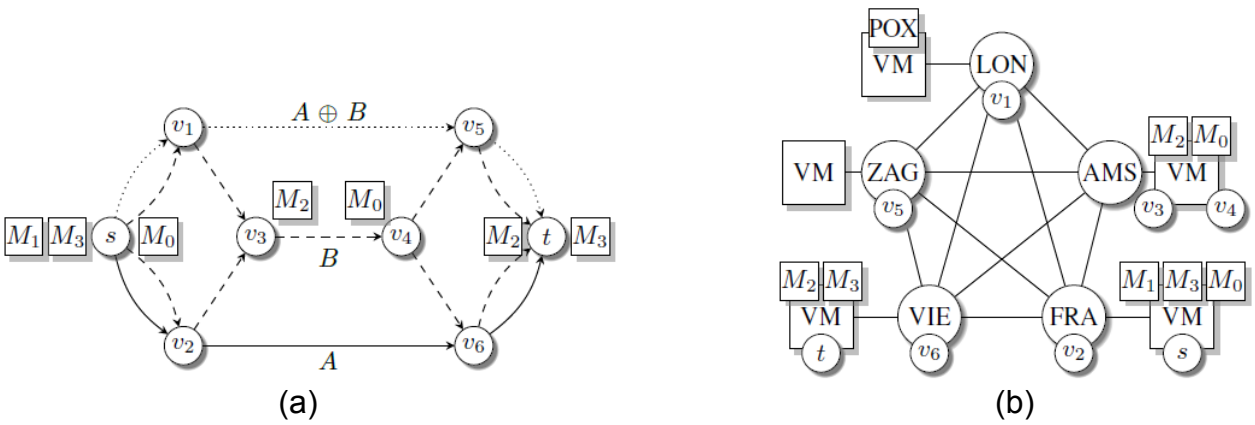


Figure 2 Experimental setup in GÉANT
The OpenFlow switches and corresponding VMs are in London, Zagreb, Vienna, Frankfurt and Amsterdam. (a) The single link failure resilient, robust network coding based solution of RFD (i.e., three end-to-end DAGs in the butterfly topology). (b) The mapping of the butterfly to the physical GOFF topology and the corresponding NF placement at the VMs.

Following the SDN/NFV principles, we implemented small building blocks as network functions (NF) allowing the creation of complex scenarios of RFD. Our implementation facilitates practical deployment of RFD in high-speed networks, such as GOFF. Even if one NF cannot process packets at line speed, the SDN controller can divide traffic among multiple NFs of the same kind running on the same virtual machine connected to the OpenFlow switch. In the following, our NF-based prototype implementation of RFD is demonstrated and possible implementation issues of the required network functions for RFD are discussed. In order to deploy an arbitrary optimal coding subgraph consisting of routing DAGs in RFD, the following types of NFs are required:

- **Splitter ($M_0$)**: The splitter duplicates the packets of its single incoming port and forwards them through two output ports. An OpenFlow switch can be programmed to follow this simple multicast rule, so no special NFs are needed (e.g., node $v_4$ duplicates the *B* flow in Figure 2).

- **Sequencer ($M_1$)**: Our RFD implementation uses MPLS labels to mark different flows (see Milestone 1.2 for details). In the RFD framework, the sequencer NF divides the user data (e.g., video stream) into parts *A* and *B* (odd and even packets in our implementation), and three MPLS labels are stacked to the packet header: the first identifies the payload type (i.e., flow *A, B* or *A XOR B*), while the other two contains the sequence number, etc., of the *A* and *B* packets in the payload, respectively. The sequencer module is always placed to the source node, e.g., *s* in Figure 2.

- **Merger ($M_2$)**: In Figure 2, the same traffic stream (*B*) arrives to node $v_3$ through two different ports. The task of the merger NF is to forward one of the packets out of the two identical copies arriving on the two incoming ports. Without loss of generality, the

flow arriving on link $(v_1, v_3)$ has lower delay, thus, in a failure-less state, the merger at $v_3$ forwards this flow and drops the duplicate packets arriving on link $(v_2, v_3)$. If a failure occurs on the path traversing $(v_1, v_3)$, then $v_3$ should forward the traffic of the intact path arriving on link $(v_2, v_3)$ instead. Note that our goal is to maintain instantaneous recovery, while the detection of the failure of path on link $(v_1, v_3)$ is never instantaneous – which results in traffic loss or increased recovery time. Thus, instead of detecting path failure, our merger NF implementation keeps track of the highest sequence number it forwarded. If a packet arrives to the merger either from link $(v_1, v_3)$ or link $(v_2, v_3)$, the merger checks whether its sequence number is higher than the current highest one. If a highest sequence number is found, then it forwards the packet and records its sequence number; otherwise dropping the packet. ***This simple forwarding logic ensures instantaneous recovery from link failures***.

- **Coding/Decoding ($M_3$)**: The simple XOR encoding allows fast packet processing, and in the network coding strategy in [Babarczi14] it is performed at the source node (*s*). From two input streams *A* and *B,* the encoder creates the *A XOR B* flow by XOR-ing the whole packet, and by restoring the checksum and other fields in the header for successful data transmission. Thus, encoding/decoding must be implemented as an NF. Note that in a general network coding (e.g., General Dedicated Protection (GDP) [Babarczi15-COMNET]) scheme, coding might be performed at intermediate nodes as well. In the case of decoding at the destination (*t*), a buffer is also necessary in order to smooth the jitter caused by the end-to-end propagation delay difference between the different subflows. Otherwise, encoding and decoding network functions are very similar.

Note that, using the above network functions, an arbitrary RFD coding subgraph can be deployed in the network.

# 4 Application Scenarios based on RFD

In order to prove the usefulness and performance of RFD in transport networks we envisioned a pair of typical applications: video streaming and distributed storage. The detailed description of the use cases and measurement results can be found in Milestone 1.3 and Milestone 1.4, respectively. The experiments are deployed on the GOFF, on top of a 5-node full-mesh topology (see Fig 1(b)), but replicated as well in different testbeds, in order to both demonstrate portability and circumvent the resource limitation of the GOFF virtual environment for high-performance scenarios. In our use cases, an SDN controller identifies or adapts a resilient topology (e.g., the butterfly in Figure 2(a)) on top of the 5-node full-mesh in GOFF (or on the testbed described in Milestone 1.4) and routes the DAGs accordingly between the nodes. The NFs are located on the VMs and connected to the network devices, thus intercepting and operating on incoming traffic as needed for each case [Ladoczki15].

## 4.1 The video streaming use case

In order to demonstrate the practical benefits of RFD for transport networks, we conducted our recovery time measurements on real equipments in a pan-European SDN-enabled network, the GÉANT OpenFlow Facility. The switches are placed at Amsterdam, Frankfurt, Vienna, London and Zagreb connected through 100 Gbps optical links. Each switch is connected to a server running a virtual machine (VM), on which the necessary network functions can be deployed. Our SDN controller is developed using POX and runs at the London VM. Besides setting appropriate flows in the switches to enable dynamic forwarding, its main objective is to map the butterfly topology in Figure 2(a) to the physical network in Figure 2(b). Each of the *A, B* and *A XOR B* subflows are identified by different forwarding identifiers and forwarded to the appropriate NFs obtained from the resilient RFD solution. For this purpose, we can use the custom network coding MPLS labels, or the VLAN ID tag of the Ethernet header for forwarding the different subflows, depending on the OpenFlow version. We placed the source node with the corresponding NFs to the Frankfurt VM, while the destination node and decoder is running at the Vienna VM.

A video flow is transmitted via UDP from the source *s* to the destination *t* through the end-to-end DAGs in Figure 2(a). By placing appropriate NFs in the topology (see Figure 2(b)), we are able to ensure instantaneous recovery after a single link failure occurring anywhere in the network as follows. The video flow in *s* (Frankfurt VM) is split into parts *A* and *B* using NF $M_1$. A coded variant *A XOR B* is also generated with $M_3$. The three flows are then routed along their corresponding DAGs, i.e., *A* is sent to the Frankfurt switch (acting as $v_2$), *A XOR B* is sent to $v_1$ in London (tunneling through the Frankfurt switch) and *B* is sent to both nodes after traversing NF $M_0$.

It is worth noting that flow *B* is routed from both Frankfurt and London switches to the Amsterdam VM ($v_3$), where both inputs are merged by $M_2$ to avoid forwarding duplicated packets. Later on, the *B* flow is split again in Amsterdam VM ($v_4$) and the DAGs are routed to the destination *t* (Vienna VM) through the Vienna switch (directly via $v_6$ and tunneling from $v_5$ in Zagreb).

In the no-failure scenario, the video is reconstructed from flows *A* and *B* by removing the custom MINERVA MPLS labels and restoring the corresponding checksums. If either *A* or *B* is broken (in RFD at most one DAG can be disrupted) the video stream can be instantaneously recovered by using packets from either flows A or B and matching them with A XOR B. Here, we present our measurement results, which were conducted on the experimental setup in Figure 2(b). Our results are the average of multiple runs with different packet sizes and different packet arrival frequencies (from ms to sec) in order to avoid false data. First, we have measured the performance of individual NFs in order to understand their performance characteristics. However, even in the most stressful scenarios, both the encoding and decoding delay was below the precision of the measurement (<= 1 ms). Thus, we can conclude that steering the traffic to the NFs at the virtual machines do not add any measurable delay to the path. Second, the end-to-end delay of the three end-to-end flows in the failure-less state was measured in the setup of Figure 2(b). Note that the *A* flow in the mapped topology traverses physical link Frankfurt-Vienna, the shortest-delay path along the edges of the *B* flow traversing physical edges Frankfurt-Amsterdam and Amsterdam-Vienna, while the *A XOR B* flow is forwarded through links Frankfurt-London, London-Zagreb and Zagreb-Vienna. The end-to-end delay of the *A, B* and *A XOR B* flows are 41.6 ms, 48 ms and 69.4 ms, respectively.

As the main goal of the RFD framework is to provide ***instantaneous recovery***, we have investigated whether the goal of *recovery time* ($t_R$) < *20-30* ms can be achieved in a real transport network [Babarczi15-COMNET]. From theory, the recovery time should be about the maximum of the pair-wise delay difference of the end-to-end paths of the original flows (*A* and *B*) and the encoded flow *A XOR B*, as after a flow disrupts it can be reconstructed from the packets of the other two. This difference is $t_R$ = *27.8* ms on average, based on the previous results. This theoretical delay was completely backed by our measurement results. We have simulated the failure of the ($v_3$, $v_4$) virtual link forwarding the *B* flow at the Amsterdam VM, and measured the recovery time at the destination node (Vienna VM), i.e., the time difference between the last packet received from the original *B* flow, and the

appearance of the first decoded *B* packet from the *A* and *A XOR B* flows. This delay was $t_R$ = *23* ms on average.

## 4.2 The distributed storage use case

RFD can also provide efficient solutions for bidirectional transmissions between a client and multiple servers, where the data is first divided into *A, B* and *A XOR B* and then stored in or recovered from different physical locations (the storage nodes). Similarly to the video streaming use case, RFD is applied to instantaneously recover data – even if one of the storages is unavailable or a link failure occurred during upload or download operation. Furthermore, and because of the distributed nature of this use case, it is possible to benefit from increased security when the three storages belong to different clouds as well (e.g., Google Drive, Dropbox, etc.), thus distributing and minimizing the risk of a potential breach of the user's stored data.

In this scenario, a user attempts to either write (upload) or read (download) some contents through a client, for instance a laptop. The client, on the other hand, interfaces with a data center consisting of a number of servers. Each server stores a specific portion of the user's data. When the user requests to upload some information, the client applies first a number of specific NFs to split and encode the data, according to the RFD architecture. Once the data is ready to be transmitted, the operation requested by the user takes place.

The SDN controller keeps track of the network status by analyzing the management traffic traversing the network –while collecting any useful information– and performs any operation related to network management. Specifically, the controller identifies first any server that joins the network (by analyzing the *registration messages* issued by any server upon booting up); to later add it to the server's pool, which is used to pick a number of available servers, as needed for each operation. The controller also keeps a pool of the current clients. Besides keeping these two pools up to date, the controller also detects the kind of user request (write/read, or upload/download) and collects any needed information for proper functioning.

When a user requests to upload a file (i.e. a write operation), the controller identifies from the server's pool those three that suit better the user request. After that, the SDN controller registers the relation between the file, server and chunk identifiers in order to be able to correctly locate the file once the user requests to download it. Prior to transmitting the file, the controller algorithm identifies three resilient paths (corresponding to the different DAGs) between the client and the servers. By setting the paths in compliance with the RFD architecture, we ensure the correct recovery of any transmission after a single-link failure. On the other hand, when the user issues a download request (i.e. a read operation), the controller finds the appropriate servers containing the chunks that correspond to the file identifier, by using the relation registered during the previous upload step. After this, the

controller redirects the different contents (i.e. data chunks) from the servers to the requesting client, where these are put back together in a similar fashion to the previous use case.

In the event of a failure in a single link (i.e. one flow is missing), the client will assemble the full contents from the received two chunks. When there is a failure condition in one of the storage nodes, the client will behave similarly. In both cases, the recovery operation of the specific file is almost immediate once both are received in the client, as there are no extra retransmissions.

# 5 Conclusions

In protection and restoration approaches without instantaneous recovery packet, retransmission and maybe flow rerouting would be necessary upon a network failure. This, however, causes severe packet loss and transport service outage; as both failure localization and failure notification times are not negligible, and the recovery time would also contain the end-to-end path delay when the flow is rerouted.

We have measured the recovery of a single working path in the GOFF after a failure disrupts the connection, and even in the best scenario it was in the order of seconds. On the other hand, in MINERVA we have shown that even if a failure occurs, our RFD approach can recover the source flow instantaneously, without packet retransmission or flow rerouting. The recovery time only depends on the delay characteristics of the underlying topology. Thus, *we have demonstrated that reducing recovery time by adding redundancy to the connection is a viable approach in transport networks*.

***We conclude that we have satisfied all requirements of the High-Availability Networking Open Call we have undertaken in the proposal***. Nevertheless, and based on the lessons learned from our work in the theoretical and implementation aspects, we see the following continuation possibilities of resilient approaches similar to MINERVA:

- **High-performance networking**: Make the RFD implementation able to handle really stressed scenarios, such as the HD video streaming investigated in Milestone 1.4 or big-sized data being reliably transmitted and stored among a number of nodes.

- **Multipath routing**: We have shown that multipath forwarding is a key enabler of RFD, and we claim that it is the basis of any future high-availability networking scheme as well. An SDN implementation of the routing on the “shortest pair of disjoint paths”, or the well-known independent tree approach would be the following steps towards a scalable and low-latency multipath routing architecture; addressed to sensitive environments, for instance clouds or 5G networks. Internet-wide multipath routing and multipath TCP are also interesting directions in this research.

- **Failure/delay monitoring**: In the MINERVA research work we have investigated all-optical failure localization, as a possible counterpart for RFD in a less time-sensitive application environment (that is, a trade-off between recovery time and resource efficiency). We have the necessary theoretical understanding of this approach (see our book published as part of MINERVA); however, we have not had any opportunity to understand its practical limitation through a real world deployment. If we can prove the existence of a common algebraic framework of all-optical monitoring and link-delay measurements, our all-optical results can be useful for link-delay monitoring in SDNs as well.

# References


**[Pasic15]** Pasic Alija, János Tapolcai, Péter Babarczi, Erika R. Bérczi-Kovács, Zoltán Király, Lajos Rónyai, Survivable Routing Meets Diversity Coding, submitted to IFIP Networking, pp. 1-9, Toulouse, France, 2015.

**[Babarczi14]** P. Babarczi, J. Tapolcai, L. Ronyai, , M. Medard, Resilient flow decomposition of unicast connections with network coding, in: Proc. IEEE International Symposium on Information Theory (ISIT), pp. 116–120, 2014.

**[Babarczi15-COMNET]** Péter Babarczi, Alija Pasic, János Tapolcai, F. Németh, and B. Ladóczki, Instantaneous recovery of unicast connections in transport networks: Routing versus coding, accepted to Elsevier Computer Networks (COMNET), impact factor 1.282 (in 2013), 2015.

**[Babarczi15-TIT]** Péter Babarczi, János Tapolcai, Alija Pasic, Lajos Rónyai, Erika R. Bérczi-Kovács, and Muriel Médard, *Linear Time Coding Algorithms for Resilient Flow Decomposition in Transport Networks*, submitted to IEEE Transactions on Information Theory, 2015.

**[Ladoczki15]** Bence Ladóczki, Carolina Fernandez, Oscar Moya, Péter Babarczi, János Tapolcai, Daniel Guija, Robust Network Coding in Transport Networks, accepted to The 34rd Annual IEEE International Conference on Computer Communications (INFOCOM), pp. 1-2, Hong Kong, 2015.

# Glossary

| | |
|---|---|
| **DAG** | Direct Acyclic Graph |
| **GDP** | General Dedicated Protection |
| **GOFF** | GÉANT OpenFlow Facility |
| **MPLS** | Multiprotocol Label Switching |
| **NFV** | Network Function Virtualization |
| **OF** | OpenFlow |
| **RFD** | Resilient Flow Decomposition |
| **SDN** | Software-Defined Networking |
| **SRDC** | Survivable Routing with Diversity Coding |
| **VLAN** | Virtual Local Area Network |
| **VM** | Virtual Machine |